COMPUTATION OF COVARIANCE MATRICES FOR CONSTRAINED PARAMETER ESTIMATION PROBLEMS USING LSQR

EKATERINA KOSTINA*, MICHAEL A. SAUNDERS†, AND INGA SCHIERLE‡

Abstract. We consider large parameter estimation problems with nonlinear equality constraints. Each Gauss-Newton iteration requires the solution of a linear least-squares problem with linear constraints. We describe the ideal numerical method based on QR factors of the constraint matrix, and show how to estimate the parameter covariance matrix. We then show how equivalent computations may be performed using the iterative least-squares solver LSQR.

Key words. constrained parameter estimation, covariance matrix of parameter estimates, optimal experimental design, nonlinear equality constraints, iterative matrix methods

AMS subject classifications. 65K10, 15A09, 65F30

1. Introduction. According to [2] we consider the constrained nonlinear parameter estimation problem

$$\begin{array}{rl} \min\limits_{x\in\mathbb{R}^n} & \frac{1}{2}||f_1(x)||_2^2 \\ \text{s.t.} & f_2(x)=0, \end{array} \tag{1.1}$$

where $f_1(x)$ is a vector of weighted residuals for a model of the form

$$\eta_i = M(x_{\text{TRUE}}, t_i) + \varepsilon_i, \qquad i = 1, \ldots, m_1. \tag{1.2}$$

Thus,

$$f_1(x) = \begin{pmatrix} (\eta_1 - M(x,t_1))/\sigma \\ \vdots \\ (\eta_{m_1} - M(x,t_{m_1}))/\sigma \end{pmatrix}, \qquad ||f_1(x)||_2^2 := \sum_{i=1}^{m_1} \frac{(\eta_i - M(x,t_i))^2}{\sigma^2}. \tag{1.3}$$

The model $M(x,t) \in \mathbb{R}$ that describes the parameter estimation problem is a nonlinear function, and $f_2(x) = 0$ is a set of equality constraints that can arise from discretization of a PDE (or ODE or DAE). In this case, the vector $x \in \mathbb{R}^n$ includes the parameters and the state variables arising from discretization, and $t \in \mathbb{R}$ is the time. We assume that measurements η_i, $i = 1, \ldots, m_1$, with measurement errors ε_i are available at times t_i, and that the measurement errors are independent and normally distributed with zero mean and variances σ.

Further we assume that $f_1 : D \subset \mathbb{R}^n \to \mathbb{R}^{m_1}$ and $f_2 : D \subset \mathbb{R}^n \to \mathbb{R}^{m_2}$ are twice-continuously differentiable and we let $J_1(x) = \frac{\partial f_1(x)}{\partial x}$ and $J_2(x) = \frac{\partial f_2(x)}{\partial x}$ denote their Jacobians. We define

$$f(x) := \begin{pmatrix} f_1(x) \\ f_2(x) \end{pmatrix}, \qquad J(x) := \begin{pmatrix} J_1(x) \\ J_2(x) \end{pmatrix}, \tag{1.4}$$

and we make the following assumptions on the problem dimensions:

*IWR, University of Heidelberg, Im Neuenheimer Feld 368, D-69120, Heidelberg, Germany (ekaterina.kostina@iwr.uni-heidelberg.de).

†Department of Management Science and Engineering, Stanford University, Stanford, CA 94305-4026 (saunders@stanford.edu). Supported by U.S. Office of Naval Research grant N00014-02-1-0076.

‡IWR, University of Heidelberg, Im Neuenheimer Feld 368, D-69120, Heidelberg, Germany (inga.schierle@gmx.de).

1. $m_2 < n$, $m_1 + m_2 \geq n$, and $n = m_2 + n_2$.
2. J_1 and J_2 satisfy the following regularity conditions on $D \subset \mathbb{R}^n$:

$$\text{rank } J_2(x) = m_2, \qquad \text{rank } J(x) = n. \tag{1.5}$$

In practice, m_1 is the number of measurements and can be moderate, m_2 comes from the PDE (or ODE or DAE) discretization and is very large, and the number of parameters is small to moderate.

To solve (1.1) we use a generalized Gauss-Newton method, in which a new iterate is generated by

$$x^{l+1} = x^l + \alpha^l \Delta x^l, \qquad 0 < \alpha^l \leq 1, \tag{1.6}$$

where Δx^l is the solution of the linearization of (1.1) at $x = x^l$:

$$\begin{array}{ll} \min\limits_{\Delta x \in \mathbb{R}^n} & \frac{1}{2}||J_1 \Delta x + f_1||^2 \\ \text{s.t.} & J_2 \Delta x + f_2 = 0, \end{array} \tag{1.7}$$

where $J_1 = J_1(x^l)$, $J_2 = J_2(x^l)$ and $f_1 = f_1(x^l)$, $f_2 = f_2(x^l)$. By an appropriate line search we get the stepsize α^l. If conditions (1.5) are fulfilled then (1.7) has a unique solution Δx^l and there exists a unique Lagrange vector λ^l that satisfies the following optimality conditions:

$$\begin{array}{rl} J_1^T J_1 \Delta x^l + J_2^T \lambda^l &= -J_1^T f_1, \\ J_2 \Delta x^l &= -f_2. \end{array} \tag{1.8}$$

We can show that Δx^l in (1.8) can be written with a solution operator J^+ as

$$\Delta x^l = -J^+(x^l) f(x^l), \tag{1.9}$$

where J^+ is a generalized inverse satisfying $J^+ J J^+ = J^+$ and given by

$$M := \begin{pmatrix} J_1^T J_1 & J_2^T \\ J_2 & 0 \end{pmatrix}, \qquad J^+ = \begin{pmatrix} I & 0 \end{pmatrix} M^{-1} \begin{pmatrix} J_1^T & 0 \\ 0 & I \end{pmatrix}. \tag{1.10}$$

Let us define

$$M^{-1} = \begin{pmatrix} J_1^T J_1 & J_2^T \\ J_2 & 0 \end{pmatrix}^{-1} := \begin{pmatrix} X & S^T \\ S & T \end{pmatrix}, \tag{1.11}$$

where $X \in \mathbb{R}^{n \times n}$, $S \in \mathbb{R}^{m_2 \times n}$, $T \in \mathbb{R}^{m_2 \times m_2}$.

Our aim is to compute a certain matrix defined by

$$C := J^+ \begin{pmatrix} I & 0 \\ 0 & 0 \end{pmatrix} (J^+)^T \in \mathbb{R}^{n \times n}. \tag{1.12}$$

It is shown in [2] that a principal submatrix of C is an approximation to the covariance matrix for parameter estimates of the constrained parameter estimation problem (1.1).

LEMMA 1.1. (from [2]). *The covariance matrix C is equal to the matrix X in (1.11) and satisfies the following linear system with respect to variables $C \in \mathbb{R}^{n \times n}$ and $S \in \mathbb{R}^{m_2 \times n}$:*

$$\begin{pmatrix} J_1^T J_1 & J_2^T \\ J_2 & 0 \end{pmatrix} \begin{pmatrix} C \\ S \end{pmatrix} = \begin{pmatrix} I \\ 0 \end{pmatrix}. \tag{1.13}$$

2. About LSQR. LSQR is an iterative method for solving large linear systems $Ax = b$ or least-squares problems $\min_x ||Ax - b||_2^2$, where $A \in \mathbb{R}^{m\times n}$ and $b \in \mathbb{R}^m$. It also returns an estimate of $\mathrm{diag}(A^TA)^{-1}$. Even if it is usually true that $m \geq n$ and rank $A = n$, these conditions need not be fulfilled. The advantage of LSQR is that it is numerically more reliable than other conjugate-gradient (CG) methods, and we can extend it to estimate more elements of $(A^TA)^{-1}$. For more information about LSQR see [4, 5].

3. Computation of C using LSQR and QR factors of J_2^T. We can rewrite (1.8) as

$$\begin{pmatrix} J_1^TJ_1 & J_2^T \\ J_2 & 0 \end{pmatrix} \begin{pmatrix} \Delta x \\ \lambda \end{pmatrix} = - \begin{pmatrix} J_1^Tf_1 \\ f_2 \end{pmatrix}, \tag{3.1}$$

with l omitted for convenience. Let us do a QR factorization of J_2^T:

$$J_2^T = Q \begin{pmatrix} R \\ 0 \end{pmatrix} \equiv \begin{pmatrix} Y & Z \end{pmatrix} \begin{pmatrix} R \\ 0 \end{pmatrix} = YR, \tag{3.2}$$

where $Y \in \mathbb{R}^{n\times m_2}$ and $Z \in \mathbb{R}^{n\times(n-m_2)}$ have orthogonal columns, $R \in \mathbb{R}^{m_2\times m_2}$ is upper triangular, and the columns of Z span the null space of J_2 (i.e., $J_2Z = 0$). Suppose

$$\Delta x = y + Zw, \tag{3.3}$$

$$\text{WHERE} \quad J_2y = -f_2. \tag{3.4}$$

The vector y is not unique, but the solution of minimum length could be obtained from $R^Tv = -f_2$, $y = Yv$, or by applying LSQR to $J_2y = -f_2$. Then w solves the problem (cf. (1.7))

$$\begin{aligned} &\min_{w\in\mathbb{R}^{n-m_2}} ||J_1(y + Zw) + f_1||^2 \\ \Rightarrow &\min_{w\in\mathbb{R}^{n-m_2}} ||(J_1Z)w - g||^2, \end{aligned} \tag{3.5}$$

where $g = -f_1 - J_1y$. We want to solve (3.5) with the help of LSQR and at the same time we want to get the covariance matrix C as a by-product of LSQR. If we modify LSQR a little, we can make it estimate some or all of $(Z^TJ_1^TJ_1Z)^{-1}$.

PROPOSITION 3.1. *The inverse of $Z^TJ_1^TJ_1Z$ exists because of assumption (1.5).*

Proof.

$$\begin{aligned} & J_2Q = J_2 \begin{pmatrix} Y & Z \end{pmatrix} = \begin{pmatrix} R^T & 0 \end{pmatrix} \\ \overset{(1.4)}{\Rightarrow} \quad & JQ = \begin{pmatrix} J_1Y & J_1Z \\ R^T & 0 \end{pmatrix}. \end{aligned} \tag{3.6}$$

Hence, J_1Z has full column rank because J has full column rank and Q is nonsingular. ∎

PROPOSITION 3.2. *The covariance matrix C satisfies*

$$C = Z(Z^TJ_1^TJ_1Z)^{-1}Z^T. \tag{3.7}$$

Proof. Referring to Lemma 1.1 we know that the j-th column of C satisfies

$$\begin{pmatrix} J_1^T J_1 & J_2^T \\ J_2 & 0 \end{pmatrix} \begin{pmatrix} C_j \\ S_j \end{pmatrix} = \begin{pmatrix} e_j \\ 0 \end{pmatrix}, \tag{3.8}$$

where C_j and S_j denote the j-th column of C and S, and e_j is the j-th column of the unit matrix I. Since we know from (3.8) that $J_2 C_j = 0$ we can write $C_j = ZU_j$, where $U_j \in \mathbb{R}^{n-m_2}$, $j = 1, \ldots, n$. Then (3.8) gives

$$\begin{aligned} & J_1^T J_1 C_j + J_2^T S_j = e_j \\ \Rightarrow \quad & Z^T (J_1^T J_1 C_j + J_2^T S_j) = Z^T e_j \\ \Rightarrow \quad & Z^T J_1^T J_1 Z U_j = Z^T e_j \\ \Rightarrow \quad & Z^T J_1^T J_1 Z U = Z^T \\ \Rightarrow \quad & C = ZU = Z(Z^T J_1^T J_1 Z)^{-1} Z^T. \end{aligned}$$

□

Now we have shown that $C = Z(Z^T J_1^T J_1 Z)^{-1} Z^T$, where we get Z from the QR factorization of J_2^T and an estimate of $(Z^T J_1^T J_1 Z)^{-1}$ from LSQR when we solve the unconstrained least-squares problem $\min ||J_1 Z w - g||$ (3.5). At the same time we get the solution Δx of the constrained problem (1.7) as follows: $\Delta x = y + Zw$ (3.3), where y comes from $J_2 y = -f_2$ (3.4) and w comes from applying LSQR to (3.5).

3.1. A more efficient way for storing the covariance matrix. Typically we don't need to estimate the whole matrix C but only the elements associated with the unknown parameters. Let $E = \{j_1, j_2, \ldots, j_{nI}\}$ be an index set that is a subset of $\{1, \ldots, n\}$, and let $\bar{I}$ be a matrix containing the corresponding columns of I. Thus, we would only compute and store $\bar{C} := \bar{I}^T C \bar{I}$.

For example: If $n = 4$ and $E = \{2, 4\}$, then

$$\bar{I}^T = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \qquad \bar{C} = \bar{I}^T C \bar{I} = \begin{pmatrix} c_{22} & c_{24} \\ c_{42} & c_{44} \end{pmatrix}.$$

In practice, at each iteration k we get $(Z^T J_1^T J_1 Z)^{-1} \approx D_k D_k^T$ from LSQR, where $D_k = \begin{pmatrix} d_1 & \ldots & d_k \end{pmatrix}$ is the set of search directions in LSQR. If we are only interested in $\bar{C} = \bar{I}^T C \bar{I} \approx (\bar{I}^T Z D_k)(\bar{I}^T Z D_k)^T$, we may update

$$\bar{C}_k = \bar{C}_{k-1} + \bar{d}_k \bar{d}_k^T,$$

where $\bar{C}_0 = 0$, $\bar{d}_k = \bar{Z} d_k$, and $\bar{Z} = \bar{I}^T Z$.

4. Computation of C with LSQR but without QR factors. In the last section we eliminated the constraints and solved the reduced problem (3.5). Since the cost of computing the QR factors of J_2^T is very high, we would like to bypass the computation of Z. To achieve this, we work with projections. Note that

$$J_2^T (J_2 J_2^T)^{-1} J_2 \overset{(3.2)}{=} Y R (R^T R)^{-1} R^T Y^T = Y Y^T \tag{4.1}$$

is the projection operator onto the subspace spanned by the columns of J_2^T, and then

$$P \equiv I - Y Y^T = Z Z^T \tag{4.2}$$

is the orthogonal projection operator onto the null space of J_2, where the matrix Z has orthogonal columns and spans the null space of J_2 (see section 3).

4.1. A modified problem. Instead of minimizing $||J_1 Zw - g||$ (3.5), we define $w = Z^T s$ and minimize $||J_1(ZZ^T)s - g|| = ||J_1 Ps - g||$. We can do so because of the following proposition.

PROPOSITION 4.1. *If $Z^T Z = I$, then solving*

$$\min_s ||J_1(ZZ^T)s - g|| \tag{4.3}$$

and setting $w = Z^T s$ is equivalent to solving $\min_w ||J_1 Zw - g||$ (3.5).

Proof. Any least-squares solution s that is a solution of (4.3) satisfies

$$\begin{aligned} & ZZ^T J_1^T J_1 ZZ^T s = ZZ^T J_1^T g \\ \overset{w=Z^T s}{\Rightarrow} \quad & ZZ^T J_1^T J_1 Zw = ZZ^T J_1^T g \\ \overset{Z^T Z=I}{\Rightarrow} \quad & Z^T J_1^T J_1 Zw = Z^T J_1^T g. \end{aligned}$$

Hence, w solves $\min_w ||J_1 Zw - g||$. Although s is under-determined, $w = Z^T s$ is unique because $J_1 Z$ has full column rank. ∎

4.2. Applying LSQR to this problem. When LSQR is applied to (4.3), the k-th iteration needs the following products for given vectors v_k and u_{k+1}:

(a) $p_1 \equiv J_1(ZZ^T)v_k$, and

(b) $p_2 \equiv (ZZ^T)J_1^T u_{k+1}$.

PROPOSITION 4.2. *For any vector $v \in \mathbb{R}^n$ the projection $Pv = ZZ^T v$ is the optimal residual $r = v - J_2^T q$ of the least-squares problem*

$$\min_q ||J_2^T q - v||. \tag{4.4}$$

Proof. If q solves (4.4), the optimal residual is

$$\begin{aligned} r \quad &= \quad v - J_2^T q \overset{(3.2)}{=} v - YRq \\ &\overset{Rq=Y^T v}{=} v - YY^T v = (I - YY^T)v = ZZ^T v. \end{aligned}$$

∎

Hence, p_1 in (a) could be obtained by solving (4.4) with $v = v_k$ and forming the product $p_1 = J_1(v_k - J_2^T q)$. By analogy, we can get p_2 in (b) by solving (4.4) with $v = J_1^T u_{k+1}$ and forming the residual $p_2 = J_1^T u_{k+1} - J_2^T q$.

Luckily p_1 can be obtained much more cheaply for each v_k.

PROPOSITION 4.3. *For each vector v_k in the solution of (4.3), $ZZ^T v_k = v_k$. Hence, $p_1 = J_1 v_k$.*

Proof. When algorithm 6 (below) is applied to (4.3), we have

$$\begin{aligned} & v_1 = ZZ^T J_1^T u_1 \\ \Rightarrow \quad & ZZ^T v_1 = Z \underbrace{Z^T Z}_{=I} Z^T J_1^T u_1 = v_1 \\ \overset{\text{by induction}}{\Rightarrow} \quad & \alpha_{k+1} ZZ^T v_{k+1} = ZZ^T J_1^T u_{k+1} - \beta_{k+1} \underbrace{ZZ^T v_k}_{=v_k} = \alpha_{k+1} v_{k+1}. \end{aligned} \tag{4.5}$$

∎

Since we want Zw (see (3.3)), note that $z \equiv Zw = ZZ^T s$, where each approximation to s from LSQR is of the form $s_k = V_k y_k$, where the columns of V_k are the

bidiagonalization vectors $v_1, \ldots, v_k$. Proposition 4.3 shows that $ZZ^TV_k = V_k$. Hence, each approximation to z is of the form $z_k = ZZ^TV_ky_k = V_ky_k = s_k$. It follows that $z = s$, the solution of (4.3). Thus, we get the solution of (1.7) as $\Delta x = y + s$, where y comes from (3.4) and s comes from (4.3). And this without using a QR factorization of J_2^T.

But we get more information from LSQR. Suppose $A \in \mathbb{R}^{m\times n}$ for any m and n (in our case $A \equiv J_1ZZ^T$). What does LSQR estimate while solving $Ax = b$? Regardless of the dimensions of A, the quantities generated by the bidiagonalization of A within LSQR (see [4]) satisfy

$$\begin{aligned} & AV_k = U_{k+1}B_k = U_{k+1}Q_k^T \begin{pmatrix} R_k \\ 0 \end{pmatrix} \\ \overset{D_k=V_kR_k^{-1}}{\Rightarrow} \quad & AD_k = U_{k+1}Q_k^T \begin{pmatrix} I \\ 0 \end{pmatrix} \\ \Rightarrow \quad & D_k^TA^TAD_k = I, \end{aligned}$$

where the columns of D_k are the directions in which the solution is updated. Even though we have "$m < n$" in our case, LSQR gives

$$\begin{aligned} \overset{A=J_1ZZ^T}{\Rightarrow} \quad & D_k^T(ZZ^TJ_1^TJ_1ZZ^T)D_k = I \\ \Rightarrow \quad & (D_k^TZ)(Z^TJ_1^TJ_1Z)(Z^TD_k) = I \\ \Rightarrow \quad & (Z^TD_k)(D_k^TZ) \approx (Z^TJ_1^TJ_1Z)^{-1}. \end{aligned}$$

The inverse of $Z^TJ_1^TJ_1Z$ exists because of assumption (1.5) (cf. section 3). We know from (3.7) that $C = Z(Z^TJ_1^TJ_1Z)^{-1}Z^T$. It follows that

$$C \approx ZZ^TD_kD_k^TZZ^T.$$

Since the columns of D_k are of the form $d_k = V_kR_k^{-1}e_k$ [4], proposition 4.3 shows that $ZZ^Td_k = ZZ^TV_kR_k^{-1}e_k = V_kR_k^{-1}e_k = d_k$. Hence,

$$C \approx D_kD_k^T,$$

where D_k is already formed during the LSQR iterations on $\min_s ||J_1(ZZ^T)s - g||$ (4.3).

4.3. A more efficient way for storing C. As before, we usually won't need all the elements of C, and it is better to compute $\bar{C}$, with

$$\bar{C} = \bar{I}^TC\bar{I} = \bar{I}^T\left(\sum d_kd_k^T\right)\bar{I} = \sum \bar{d}_k\bar{d}_k^T,$$

where $\bar{d}_k = \bar{I}^Td_k$.

5. Summary of solution procedure. The steps for solving (1.7) for $(\Delta x, \lambda)$ can now be summarized as follows. Let λ^l be the current approximation to λ.

1. Solve $J_2y = -f_2$.
2. Form $g = -f_1 - J_1y$.
3. Solve $\min_s ||J_1ZZ^Ts - g||$, obtaining s and possibly estimates of a submatrix of C.
4. Set $\Delta x = y + s$.
5. Form $\bar{g} = g - J_1s$ and $h = J_1^T\bar{g} - J_2^T\lambda^l$.
6. Solve $\min_{\Delta\lambda} ||J_2^T\Delta\lambda - h||$.
7. Set $\lambda = \lambda^l + \Delta\lambda$.

6. LSQR with covariance estimation. The following steps summarize LSQR applied to the problem $\min_x ||Ax - b||$ (4.3). The output includes specified rows and columns of C.

Input **ind** = specified index set
Step 1 *(Initialization)*
$\beta_1 u_1 = b, \quad \alpha_1 v_1 = A^T u_1, \quad y_1 = v_1, \quad d_0 = 0, \quad x_0 = 0,$
$\bar{\phi}_1 = \beta_1, \quad \bar{\rho}_1 = \alpha_1, \quad var = 0, \quad \mathbf{cov = 0}$
Step 2 For $k = 1, 2, 3, \ldots$ repeat steps **2.1**–**2.4**.
2.1 *(Continue the bidiagonalization)*
1. $\beta_{k+1} u_{k+1} = A v_k - \alpha_k u_k$
2. $\alpha_{k+1} v_{k+1} = A^T u_{k+1} - \beta_{k+1} v_k$

2.2 *(Construct and apply next orthogonal transformation)*
1. $\rho_k = (\bar{\rho}_k^2 + \beta_{k+1}^2)^{\frac{1}{2}}$
2. $c_k = \bar{\rho}_k / \rho_k$
3. $s_k = \beta_{k+1} / \rho_k$
4. $\theta_{k+1} = s_k \alpha_{k+1}$
5. $\bar{\rho}_{k+1} = -c_k \alpha_{k+1}$
6. $\phi_k = c_k \bar{\phi}_k$
7. $\bar{\phi}_{k+1} = s_k \bar{\phi}_k.$

2.3 *(Update x, y and cov)*
1. $d_k = (1/\rho_k) y_k$
2. $var = var + d_k .* d_k$
3. $\mathbf{cov = cov + d_k(ind) d_k(ind)^T}$
4. $x_k = x_{k-1} + (\phi_k / \rho_k) y_k$
5. $y_{k+1} = v_{k+1} - (\theta_k / \rho_k) y_k.$

2.4 *(Test for convergence)*
For stopping criteria we refer to [4].

Only the variables/lines in bold letters were added compared to the standard LSQR version. The output matrix **cov** is a generalization of the vector $var = \sum \mathrm{diag}(d_k d_k^T)$ normally computed by LSQR. All the other variables are also needed for the standard LSQR version.

7. Numerical results. To test algorithm 6 for computing the covariance matrix for parameter estimates using LSQR, we have implemented it in MATLAB. We investigate the effect of choosing different accuracies for the computation of the projections by solving equation (4.4) by means of LSQR. And, for numerical comparisons, we also compute the covariance matrix by equation (3.7) in MATLAB. The results of algorithm 6 should approach these values.

PRACTICAL APPLICATION OF ALGORITHM: The projections required by algorithm 6 are computed by LSQR, where we use different accuracies as stopping criterion for the computation of the projections. Here, we use a diagonal matrix $D_{pr} = \mathrm{diag}(1/\delta_i)$ as a simple preconditioner, where δ_i is the norm of the i-th row of J_2. This means that if we compute the projections $Pv = ZZ^T v$ by LSQR then we use D_{pr} as right-preconditioner and apply LSQR to

$$\min_{q_D} ||J_2^T D_{pr} q_D - v||.$$

Then $q = D_{pr} q_D$ and the projection is again $Pv = v - J_2^T q$.

Let us call the LSQR iterations in algorithm 6 *"outer"* LSQR *iterations*. And if we compute projections Pv by another call of LSQR we call these iterations *"inner"*

	Direct method based on QR factors using equation (3.7)	**Iterative method** based on LSQR for solving $\min_s ||J_1(ZZ^T)s - g||$. Stopping criterion for these "outer" iterations: `Atol = btol =` **1e-08**. **Computation of projections** by using LSQR with accuracy `Atol = btol =`			
		1e-08	1e-10	1e-12	1e-14
p_1	1.3795e-03	3.3135e-02	1.3799e-03	1.3795e-03	1.3795e-03
p_2	1.9487e-03	7.0117e-03	1.9472e-03	1.9487e-03	1.9487e-03
p_3	2.6670e-03	3.9042e-03	2.6602e-03	2.6671e-03	2.6670e-03
p_4	1.7011e-03	7.4479e-02	1.7013e-03	1.7011e-03	1.7011e-03
p_5	1.4812e-03	1.4323e-02	1.4767e-03	1.4813e-03	1.4812e-03
p_6	2.5454e-03	6.2793e-02	2.5363e-03	2.5456e-03	2.5454e-03
p_7	1.1830e-03	1.1058e-02	1.1771e-03	1.1832e-03	1.1830e-03
p_8	3.7684e-03	2.2724e-02	3.7712e-03	3.7683e-03	3.7684e-03
p_9	1.4075e-03	1.8234e-03	1.4086e-03	1.4075e-03	1.4075e-03
p_{10}	3.3090e-03	5.7392e-03	3.3262e-03	3.3088e-03	3.3090e-03
"outer" LSQR iterations:	-	72	6	6	6

TABLE 7.1

Last ten elements C_{ii} of C computed with a direct method based on QR factors and with the iterative method LSQR.

LSQR *iterations*. The matrix D_{pr} can also be used to do left-preconditioning on (3.4) by solving $D_{pr}J_2y = -D_{pr}f_2$ instead of (3.4).

REPRESENTATION OF THE RESULTS: The tables of results are structured as follows: The first column names the parameters for which the diagonal elements C_{ii} are computed. The second column gives the results when computing the diagonals C_{ii} by equation (3.7). The next three or four columns give the results for computing the diagonals C_{ii} by the iterative method LSQR, where the projections are also computed with LSQR but with different accuracies (from 1e-8 up to 1e-14). The last row gives the required number of "outer" LSQR iterations for achieving a specified accuracy when solving equation (4.3). The computations were performed using the following hardware and software configuration:

- Intel Centrino 1.4 GHz,
- 512 MB RAM,
- MATLAB Version 7.1.0.124(R14) for Microsoft Windows XP.

7.1. First example: Random matrices and random vectors. In this subsection we consider an example where we generate dense random matrices for J_1, J_2 and random vectors for f_1, f_2 by MATLAB. We have

- $\text{cond}(J_1) \approx 4.5604\text{e+}01$,
- $\text{cond}(J_2) \approx 2.3596\text{e+}03$,

where J_1 is a 120×326 matrix and J_2 is a 320×326 matrix. This example fulfills the regularity conditions (1.5). In Table 7.1 we compare the results of computing the diagonals C_{ii} in different ways and we only give the last ten diagonal elements of C.

7.2. Second example: Elbow robot. In this case we apply algorithm 6 to a discretized boundary value problem (BVP) with collocation used as a discretization method [1]. For a fuller description of the model of the elbow robot we refer to [6]. We were estimating 7 parameters and data was simulated by adding normally distributed

	Direct method based on QR factors using equation (3.7)	**Iterative method** based on LSQR for solving $\min_s \|\|J_1(ZZ^T)s - g\|\|$. Stopping criterion for these "outer" iterations: `Atol = btol =` **1e-10**. **Computation of projections** by using LSQR with accuracy `Atol = btol =` 1e-10	1e-12	1e-14
p_1	1.2901e+02	1.2742e+02	1.2688e+02	1.2674e+02
p_2	2.5318e+02	2.3386e+02	2.3326e+02	2.3289e+02
p_3	6.8045e-02	7.2075e-02	7.1810e-02	7.1773e-02
p_4	2.4214e+02	2.3849e+02	2.3361e+02	2.3260e+02
p_5	3.7778e+02	3.2440e+02	3.2293e+02	3.2224e+02
p_6	4.7040e+00	5.4169e+00	5.2093e+00	5.3135e+00
p_7	3.5700e-01	3.2167e-01	3.2036e-01	3.1978e-01
"outer" LSQR iterations:	-	21	18	17

TABLE 7.2

Diagonals C_{ii} of the parameter estimates computed with a direct method based on QR factors and with the iterative method LSQR.

noise. We have

- $\text{cond}(J_1) = 1$,
- $\text{cond}(J_2) \approx 6.8706\text{e+}05$,

where J_1 is a 246×973 matrix and J_2 is a 960×973 matrix.

In Table 7.2 we compare the results of computing the diagonals C_{ii} of the parameter estimates in different ways.

7.3. Summing up the results. One can see that it is important to compute the projections (see equation (4.4)) with a high precision. This is plausible because when LSQR is applied to $\min_s ||J_1(ZZ^T)s - g||$ (4.3) it needs the following products for given vectors v_k and u_{k+1} in every iteration:

(a) $p_1 := J_1(ZZ^T)v_k$, and
(b) $p_2 := (ZZ^T)J_1^T u_{k+1}$.

To (a): In proposition 4.3 we have shown that $ZZ^T v_k = v_k$. But this is only true with exact arithmetic and holds by approximation if the projections are computed with high accuracy.
To (b): In order that LSQR still provides a good approximate solution to (4.3) the product $(ZZ^T)J_1^T u_{k+1}$ must be quite accurate because in the derivation of the theory we did not take rounding errors and their impact on the solution into account.
Furthermore, we have to be aware of only having an estimate of the solution y of equation (3.4) $J_2^T y = -f_2$ that is required as input for algorithm 6 (see equation (3.5)). In the last Gauss-Newton iteration we can take $y = 0$ for computing parts of C. Note, for such "small" problems it is usually better to compute the covariance matrix with direct methods. We have chosen these "smaller" examples for better numerical comparisons of iterative and direct methods.

8. Summary and outlook. In this paper, we have obtained an adequate representation of the covariance matrix based on the iterative least-squares method LSQR. The generalized Gauss-Newton method that we apply to solve the constrained finite nonlinear optimization problem requires the solution of a linear least-squares problem with linear constraints in each iteration. We have shown that if we solve these

least-squares problems with LSQR then we can estimate the diagonals of C and small principal submatrices of C at essentially no cost. Note, since we do not need C for every iteration step l of the Gauss-Newton iteration, we only have to compute C at the last iteration. Now, let us say a few words about practical applications and forthcoming research topics.

Computation of projections: One of the main problems is that LSQR requires the product $p_2 := (ZZ^T)J_1^T u_{k+1}$ for a given vector u_{k+1} in each iteration k. Thus, if we compute the projection $PJ_1^T u_{k+1} = (ZZ^T)J_1^T u_{k+1}$ by means of LSQR then we have to solve another least-squares problem for every iteration k in LSQR (see (4.4)). Depending on the condition number and the singular value clustering of $J_1 ZZ^T$ this can be very expensive. If a basis Z for the null space of J_2 were given directly we could bypass these "inner" iterations for computing $ZZ^T J_1^T u_{k+1}$ by LSQR and compute C by equation (3.7), where would get an estimate of $\left(Z^T J_1^T J_1 Z\right)^{-1}$ from LSQR when solving (3.5). But since in general Z is a dense matrix the explicit use of Z will cause the iterations to be expensive. Thus, it is better to use approaches that bypass the computation of Z. As mentioned in [3], the price we have to pay for these alternatives is that they can cause excessive rounding errors that can result in slowing down the optimization process or even preventing it from converging.

Preconditioning: If J_2 is ill-conditioned then right-preconditioning of some kind is recommended when computing the projections by LSQR (or by some other method). Thus, we want to get a more favorable distribution of the eigenvalues of J_2 to reduce the required number of iterations until convergence. Left-preconditioning should not be used for least-squares problems because then the problem would be altered. Note that right-preconditioning with a preconditioner D_{pr} on problem (4.3) doesn't bring any advantages because $J_1 ZZ^T D_{pr} v_k \neq J_1 v_k$ and we would lose the property $ZZ^T v_k = v_k$. This means we would have to compute $J_1 ZZ^T D_{pr} v_k$ for every iteration k by solving another least-squares problem with LSQR. Furthermore we would not get the covariance matrix as easily as without preconditioning.

Future research topics: Summing up, further research on iterative methods for parameter estimation and design of optimal parameters in dynamic processes should be devoted to

- numerical aspects including the effective implementation of the methods,
- the choice of effective preconditioners, and
- the numerical computation of the derivatives of the covariance matrix. Then the results of this paper can be applied to optimum experimental design methods.

It would also be interesting to apply this new method to real problems, where the dynamic processes are modeled by PDEs.

REFERENCES

[1] H.-G. Bock, *Randwertproblemmethoden zur Parameteridentifizierung in Systemen nichtlinearer Differentialgleichungen*, Universität Bonn, Bonner Mathematische Schriften 183 (1987), Bonn.

[2] H.-G. Bock, E. Kostina, and O. Kostyukova, *Covariance matrices for parameter estimates of constrained parameter estimation problems*, SIAM J. on Matrix Analysis and Applications 29(2) (2007), pp. 626–642.

[3] N. I. M. Gould and M. E. Hribar and J. Nocedal, *On the solution of equality constrained quadratic programming problems arising in optimization*, SIAM J. on Scientific Computing 23(4) (2001), pp. 1375 – 1394.

[4] C. C. Paige and M. A. Saunders, *LSQR: An algorithm for sparse linear equations and sparse least-squares*, ACM Trans. Math. Softw. 8(1) (1982), pp. 43–71.

[5] C. C. Paige and M. A. Saunders, *LSQR: Sparse linear equations and least-squares*, ACM Trans. Math. Softw. 8(2) (1982), pp. 195–209.

[6] I. Schierle, *Iterative Methods for the Computation of Covariance Matrices for Large-Scale Constrained Nonlinear Parameter Estimation Problems*, Diplomarbeit, Universität Heidelberg (2008).